

UNIVERSITÀ DEGLI STUDI DI PISA

Facoltà di Ingegneria

Corso di Laurea in Ingegneria Elettronica

Tesi di Laurea

Studio, progetto e realizzazione di un analizzatore di
traffico su rete CAN

Relatore:
Prof. Roberto Saletti

Candidato:
Giuseppe Tognini

Relatore:
Prof. Roberto Roncella

ANNO ACCADEMICO 2002-2003
3 dicembre 2003

Le tesi possono essere lavori più o meno originali che possono richiedere più o meno tempo, possono essere più o meno interessanti per chi le legge e per chi le scrive ... un'unica cosa è certa: non sono mai frutto del lavoro e dell'impegno di una sola persona. Per questo motivo voglio ringraziare il mio compagno di sventura in questa carriera universitaria e futuro ingegnere informatico, Manuele Piastra senza il cui aiuto sarei naufragato nei meandri del Visual C++ e del T_EX; ringrazio l'ing. Federico Baronti per i suoi preziosi consigli, il prof. Roberto Saletti, l'ing. Carlo Napoli, l'ing. Diego Lunardini e la mia ragazza per la pazienza che hanno avuto in tutti questi mesi e soprattutto ringrazio la mia famiglia che mi ha dato questa l'opportunità.

Indice

1	Protocolli di comunicazione industriali	2
1.1	Il modello a strati	2
1.1.1	Il modello ISO/OSI (OSI reference model)	2
1.1.2	Pila e protocolli	5
1.1.3	CAN, HLPs e profiles	7
1.2	I bus di campo o Fieldbus	10
1.2.1	Cos'è una Fieldbus Network?	10
1.2.2	Perché utilizzare una Fieldbus Network	10
1.2.3	Le tecnologie	12
1.2.4	Caratteristiche a confronto	13
1.3	I protocolli di alto livello del CAN	14
1.3.1	Caratteristiche comuni	14
1.3.2	OSEK/VDX	16
1.3.3	SAE J1939	17
1.3.4	CAL	19
1.3.5	CANopen	21
1.3.5.1	Object Dictionary	22
1.3.5.2	Tipi di messaggi	22
1.3.5.3	Predefined Connection Set	24
1.3.6	DeviceNet	24
1.3.6.1	CIP e tipi di messaggi	25
1.3.6.2	Caratteristiche	26
1.3.7	SDS	27
1.3.8	CANKingdom	28
1.4	Le applicazioni del CAN	29
1.4.1	Automotive	29
1.4.2	Strumenti elettromedicali	31
1.4.3	Applicazioni industriali	32

2	Gli analizzatori di traffico	34
2.1	Che cos'è un analizzatore di traffico	34
2.2	DAQ e CAN in applicazioni automotive	35
2.3	Tipi di analizzatori	36
2.4	Descrizione dello stato dell'arte	36
2.4.1	Warwick Control Technology	40
2.4.1.1	X-Analyser	40
2.4.1.2	X-Analyser CANpack	42
2.4.1.3	X-Analyser REMOTE	42
2.4.2	Intrepid Control Systems	43
2.4.2.1	Vehicle Spy	43
2.4.2.2	neoVI	44
2.4.3	Vector Informatik GmbH	46
2.4.4	Architettura dell'interfaccia hardware	47
3	Informazioni preliminari	51
3.1	Introduzione al CAN	51
3.1.1	Accesso al bus e struttura dei frame	52
3.1.1.1	Struttura dei frame	53
3.1.1.2	CSMA/CD	60
3.1.1.3	Non-destructive bitwise arbitration	61
3.1.2	Sequenza CRC	65
3.1.2.1	Concetti base	65
3.1.2.2	Calcolo del CRC	66
3.1.2.3	Errori rilevati da CRC	67
3.1.2.4	La sequenza CRC del protocollo CAN	68
3.1.3	Gestione degli errori	69
3.1.3.1	Cause di errore	70
3.1.3.2	Condizioni di errore	70
3.1.3.3	Fault confinement	72
3.1.4	Codifica dei messaggi	74
3.1.4.1	La codifica NRZ	74
3.1.4.2	Bit-stuffing	75
3.1.5	CAN physical layer	75
3.1.6	Bit timing	78
3.1.6.1	Il bit time di una rete CAN	78
3.1.6.2	I quattro segmenti del bit time	80
3.1.6.3	Programmazione dei bit-time segment	80
3.1.6.4	Istante di campionamento	81
3.1.6.5	Hard Synchronization	81
3.1.6.6	Risincronizzazione	82

3.1.7	Impostazione dei parametri	85
3.1.7.1	Il Propagation Delay	85
3.1.7.2	Oscillator Tolerance	87
3.1.7.3	Tolleranza massima ammissibile	88
3.1.7.4	Considerazioni sul Baudrate	92
3.2	Informazioni sul sistema di sviluppo utilizzato	95
3.2.1	Introduzione	95
3.2.2	L'ambiente di sviluppo	96
3.2.3	Implementazione della rete CAN di test	98
4	<i>CANinterface</i>	102
4.1	L'interfaccia hardware <i>CANinterface</i>	102
4.1.1	Il microcontrollore ATMEL T89C51CC01	103
4.1.1.1	Il circuito di generazione del clock	108
4.1.1.2	Il circuito di reset	111
4.1.1.3	Attivazione della modalità di programmazione	112
4.1.2	La memoria	113
4.1.3	Il transceiver Philips PCA82C250	114
4.1.4	L'interfaccia di collegamento RS232 con il PC	117
4.1.5	Il circuito di alimentazione	120
4.2	<i>CAN2USBinterface</i> : una nuova interfaccia seriale	120
4.2.1	USB e RS232 a confronto	123
4.2.2	Caratteristiche del convertitore FT8U232BM	123
4.2.3	Come modificare <i>CANinterface</i>	124
4.2.3.1	Modifiche hardware	124
4.2.3.2	Modifiche software	127
5	Il software di <i>CANanalyser</i>	129
5.1	<i>MICROsoftware</i>	129
5.1.1	Struttura ed elenco dei comandi	130
5.1.1.1	I comandi inviati da <i>CANsoftware</i>	130
5.1.1.2	I comandi inviati da <i>MICROsoftware</i>	135
5.1.2	Dati inviati al PC durante l'acquisizione: "CAN messages"	137
5.1.2.1	"CodaRX": struttura ed utilizzo	137
5.1.2.2	Struttura dei "CAN message"	139
5.1.2.3	Gestione dell'overflow	141
5.1.2.4	Gestione del "time stamp"	143
5.1.3	Le altre procedure	147
5.1.3.1	La porta di comunicazione seriale	148
5.1.3.2	Programmazione della UART	148

5.1.3.3	Rilevamento del CAN bit-rate in automatico .	149
5.2	CAN <i>software</i>	150
5.2.1	L'architettura documento/vista	152
5.2.2	La classe "CCanDATA"	155
5.2.2.1	Le variabili membro	157
5.2.2.2	Le funzioni membro	160
5.2.3	CAN <i>software</i> : applicazione multithread	162
5.2.3.1	La classe "CCoda"	162
5.2.3.2	Task 3: la gestione della seriale	166
5.2.3.3	Task 2: costruzione degli oggetti CAN	167
5.2.3.4	Task 1: visualizzazione dei dati	171
5.2.4	Menu: "setup"	173
5.2.4.1	Parametri di bit timing	175
5.2.4.2	Filtro dei messaggi	178
5.2.4.3	Gestione dei "segnali"	179
5.2.4.4	Parametri di funzionamento dell'applicazione .	184
5.2.5	Menu: "action"	184
5.2.5.1	Modalità di "Listening"	186
5.2.5.2	Modalità di "Playback"	186
5.2.6	Menu: "file"	186
5.2.6.1	Salvataggio dei dati	187
5.2.6.2	Esportazione dei dati	187
5.2.7	La barra di stato	188
6	Conclusioni	192
6.1	Test dell'analizzatore	192
6.2	Performance	198
6.3	Limiti	201
6.4	Note dell'autore	203
A	Configurazione hardware dell'ATMEL T89C51CC01	205
A.1	Configurazione dei pin	205
A.2	Circuito interno di generazione del clock	206
A.3	Circuito interno di generazione del reset	206
B	ISP e flash memory del T89C51CC01	210
B.1	FM0 e FM1	210
B.2	In-System Programming (ISP)	211
C	Gestione dei parametri di bit timing nel T89C51CC01	216

D	Lo standard OBD-II	220
E	Diffusione del CAN nei sistemi di diagnostica on-board	224
F	Formati dei bit	229
F.1	Il campo “Format Type”	229
F.2	Il campo “Data Type”	229
G	Calcolo del “WorkLoad”	232
G.1	“WorkLoad” istantaneo	232
G.2	“WorkLoad” medio	234
H	Struttura dei file esportati	235
H.1	Esportazione in formato “.csv”	235
H.2	Esportazione in formato “.dif”	236
H.2.1	Organizzazione di un file “.dif”	238
H.2.2	Header	238
H.2.3	Body	239
I	Codice di MICRO<i>software</i>	240
I.1	File: “t89c51cc01.h”	240
I.2	File: “compiler.h”	248
I.3	File: “can_drv.h”	250
I.4	File: “can_drv.c”	251
I.5	File: “autobaud.a51”	255
I.6	File: “config.h”	259
I.7	File: “can_lib.h”	260
I.8	File: “can_lib.c”	264
I.9	File: “serial_drv.h”	271
I.10	File: “serial_drv.c”	271
I.11	File: “main.c”	275

Elenco delle figure

1.1	OSI/ISO reference model	3
1.2	Suddivisione in layer di due sistemi connessi	6
1.3	Modello funzionale di uno <i>strato</i>	6
1.4	Layer, standard e implementazione di un bus CAN.	8
1.5	CANopen: communication profiles e device profiles.	9
1.6	Analogia tra linguaggio e protocolli di alto e basso livello.	9
1.7	Esempio di sistema in rete a controllo distribuito ¹⁰	11
1.8	Tabella riepilogativa delle caratteristiche di più diffusi protocolli “open” per Fieldbus Networks ¹³ . [6]	15
1.9	Esempio applicativo del protocollo SAE J1939.	18
1.10	CANopen Object Dictionary	21
1.11	Struttura del campo CAN-Identifier nel CANopen-Predefined Connection Set.	24
1.12	DeviceNet network	26
1.13	Struttura del campo CAN-Identifier nel DeviceNet-Predefined Master/Slave Connection Set.	27
1.14	SDS: struttura del campo CAN-Identifier.	28
1.15	CANKingdom: rappresentazione convenzionale di una rete CAN.	29
1.16	Schematizzazione di una rete CAN impiegata in campo automotive.	30
1.17	Schematizzazione di una rete CAN impiegata per la realizzazione di un sistema <i>X-Ray angio-biplane</i>	31
2.1	Schema funzionale a blocchi di un analizzatore di traffico per rete CAN.	37
2.2	Schema funzionale a blocchi della CAN interface	38
2.3	Architettura di un CAN Controller di tipo <i>Stand-alone</i>	40
2.4	Warwick Control Technology: X-Analyser	41
2.5	Warwick Control Technology: utilizzo di X-Analyser REMOTE per l’analisi remota di un sistema CAN.	42
2.6	Intrepid Control Systems: Vehicle Spy	43

2.7	Vector Informatik CANalyser	46
2.8	ESD: schema funzionale a blocchi dell'interfaccia "CAN-USB Mini Interface".	48
2.9	ESD: Schema funzionale a blocchi dell'interfaccia "CAN - Bluetooth".	49
3.1	Struttura schematica di una rete CAN.	52
3.2	Formato dei DATA e REMOTE frame - Formato standard (CAN1.2).	54
3.3	Formato dei DATA e REMOTE frame - Formato esteso (CAN-2.0).	55
3.4	Struttura dell'ERROR Frame.	58
3.5	Struttura dell'INTERFRAME Space.	59
3.6	Struttura dell'OVERLOAD Frame.	59
3.7	Comportamento del nodo CAN al rilevamento di errori nei campi di EOF e Intermission Space.	60
3.8	Meccanismo del Wired-AND.	62
3.9	Diagramma a blocchi esplicativo delle operazioni eseguite da ogni nodo CAN durante la fase di arbitraggio del bus.	63
3.10	Meccanismo dell'arbitraggio <i>bitwise non distruttivo</i> : (A) I nodi 1, 2 e 3 iniziano contemporaneamente la trasmissione del messaggio, il nodo 3 rileva un bit dominante, sospende la trasmissione ed entra in ricezione. (B) Anche il nodo 1 entra in ricezione. (C) Il nodo 2 vince l'arbitraggio e <i>continua</i> l'invio del frame, il nodo 1 filtra l'ID e lo accetta, il nodo 3 lo rifiuta e non elaborerà i dati trasmessi da 2.	64
3.11	Calcolo del CRC: divisione binaria modulo 2 tra $x^RT(x)$ e $G(x)$	67
3.12	"CRC Area" e coefficienti del polinomio di calcolo $M(x)$	68
3.13	Sezioni del frame su cui il CAN verifica la presenza di Form-Error.	71
3.14	Sezioni del frame su cui il CAN verifica la presenza di Bit-Error.	72
3.15	Regole di <i>fault confinement</i> dei nodi di una rete CAN.	73
3.16	Codifica dei bit mediante l'uso della tecnica NRZ.	74
3.17	Metodo del bit-stuffing.	76
3.18	ISO11898-2 physical layer.	77
3.19	ISO 11898-3 e ISO 11992 physical layer.	78
3.20	Schematizzazione del processo di determinazione del <i>bit time</i>	79
3.21	Esemplificazione della procedura di sincronizzazione tra due nodi di una generica rete.	82
3.22	Sincronizzazione dei nodi: errore di fase nullo.	83
3.23	Sincronizzazione dei nodi: errore di fase positivo.	84

3.24	Sincronizzazione dei nodi: errore di fase negativo.	84
3.25	Esemplificazione di un arbitraggio errato per effetto del <i>Propagation Delay</i>	86
3.26	Ritardo introdotto sulla propagazione del segnale <i>Propagation Delay</i>	87
3.27	Valutazione dello shift di fase tra due nodi nel caso peggiore (worst case).	88
3.28	Local Error in presenza di almeno due nodi Error Active.	89
3.29	Local Error in presenza di nodi tutti passivi.	90
3.30	92
3.31	Costruzione del bit time per ottenere la condizione di <i>Maximum Oscillator Tolerance</i>	93
3.32	Bit timing per ottenere <i>Maximum Bit Rate</i>	94
3.33	Variazione del bitrate in funzione della lunghezza del bus.	94
3.34	Schema funzionale del ciclo di sviluppo del software per mezzo del software μ Vision2.	98
3.35	Rete CAN realizzata per il test del sistema.	99
3.36	Processo di acknowledgment check del messaggio in una rete CAN.	100
4.1	ATMEL T89C51CC01: schema funzionale a blocchi.	103
4.2	ATMEL T89C51CC01: mailbox ed gestione della memoria tramite registri SFR.	105
4.3	Esempio di configurazione di un CAN buffer.	106
4.4	CAN <i>interface</i> : schema di collegamento del microcontrollore ATMEL T89C51CC01.	109
4.5	CAN <i>interface</i> : schema del circuito di generazione del clock.	111
4.6	CAN <i>interface</i> : schema del circuito di generazione del reset.	112
4.7	CAN <i>interface</i> : circuito di gestione delle modalità di programmazione/esecuzione.	112
4.8	ATMEL T89C51CC01: organizzazione della memoria RAM.	114
4.9	Philips PCA82C250: schema funzionale a blocchi.	116
4.10	CAN <i>interface</i> : schema di collegamento del transceiver Philips PCA82C250.	116
4.11	Philips PCA82C250: diagramma di temporizzazione del transceiver.	117
4.12	Maxim MAX232: schema funzionale a blocchi.	118
4.13	CAN <i>interface</i> : schema di collegamento del Maxim MAX232.	119
4.14	CAN <i>interface</i> : circuito di alimentazione.	121
4.15	CAN <i>interface</i> : schema completo.	122
4.16	USBMOD1: schema del modulo prodotto dalla "Raven".	125

4.17	CAN2USB <i>interface</i> : schema funzionale.	126
5.1	MICRO <i>software</i> : descrizione dell'handshake tra CAN <i>software</i> e MICRO <i>software</i>	131
5.2	ATMEL T89C51CC01: meccanismo di filtraggio degli identificatori o "Acceptance Filtering".	133
5.3	MICRO <i>software</i> : dettaglio della struttura del comando di over-buffer.	136
5.4	ATMEL T89C51CC01: diagramma semplificato a blocchi dell'utilizzo dei registri CANTIM e CANSTMP.	136
5.5	MICRO <i>software</i> : struttura ed organizzazione di "CodaRX".	138
5.6	MICRO <i>software</i> : formattazione ed organizzazione dei "CAN message".	140
5.7	MICRO <i>software</i> : sincronizzazione dei "CAN message" l'invio del comando di overrun.	144
5.8	MICRO <i>software</i> : overrun in condizione di "CodaRX" vuota.	146
5.9	MICRO <i>software</i> : overrun in condizione di "CodaRX" piena.	147
5.10	ATMEL T89C51CC01: timing dei segnali della UART configurata in modalità 1.	149
5.11	MICRO <i>software</i> : diagramma di flusso a blocchi della procedura di "autobaud()".	151
5.12	Applicazioni SDI e MDI: gerarchia delle finestre	152
5.13	CAN <i>software</i> : architettura dell'interfaccia.	153
5.14	CAN <i>software</i> : viste dell'applicazione.	156
5.15	MICRO <i>software</i> : esemplificazione dell'interazione tra gli oggetti "CCanDATA" ed una delle viste (CCanDataView).	157
5.16	CAN <i>software</i> : schematizzazione delle strutture dati condivise tra i task e delle operazioni che interagiscono con esse.	163
5.17	CAN <i>software</i> : implementazione di "CCoda"	165
5.18	Esempio di funzionamento delle primitive semaforiche: i task A, B, C e D condividono la stessa risorsa per mezzo di un semaforo che consente l'accesso contemporaneo alla risorsa ad un massimo di 3 task (semaforo inizializzato a 3).	166
5.19	CAN <i>software</i> : diagramma di flusso a blocchi della funzione "run()".	168
5.20	CAN <i>software</i> : task di costruzione degli oggetti "CCanDATA".	169
5.21	CAN <i>software</i> : meccanismo di visualizzazione dei dati attraverso le 4 viste disponibili.	172
5.22	MICRO <i>software</i> : visualizzazione dei messaggi.	173
5.23	MICRO <i>software</i> : visualizzazione dei "segnali".	174
5.24	MICRO <i>software</i> : visualizzazione grafica dei "segnali".	174

5.25	MICRO <i>software</i> : visualizzazione delle statistiche.	175
5.26	CAN <i>software</i> : finestra per l'impostazione dei parametri di bit timing della rete CAN.	176
5.27	CAN <i>software</i> : tabella riepilogativa dei parametri di configurazione.	177
5.28	CAN <i>software</i> : finestra di impostazione dei parametri di filtro dei messaggi.	178
5.29	CAN <i>software</i> : finestra di editing dei segnali.	180
5.30	CAN <i>software</i> : visualizzazione dei segnali.	181
5.31	CAN <i>software</i> : finestra di impostazione dei parametri.	185
5.32	CAN <i>software</i> : menu "file".	187
5.33	CAN <i>software</i> : esempio di un file ".dif" generato da CAN <i>software</i> ed aperto con <i>Microsoft Windows EXCEL</i> ©.	189
5.34	CAN <i>software</i> : barra di stato dell'applicazione.	190
6.1	Rete CAN realizzata per il test del sistema.	193
6.2	Schema di generazione dei burst utilizzati per il test.	195
6.3	Rete CAN realizzata per la seconda serie di test del sistema.	196
6.4	Visualizzazione dei dati acquisiti durante una delle prove della seconda serie di test.	197
6.5	Visualizzazione della condizione di "overflow".	202
A.1	ATMEL T89C51CC01: configurazione dei pin.	205
A.2	ATMEL T89C51CC01: parte interna al microcontrollore del circuito di oscillazione per la generazione del clock di sistema.	206
A.3	ATMEL T89C51CC01: parte interna al microcontrollore del circuito di generazione del reset.	207
B.1	ATMEL T89C51CC01: organizzazione della memoria flash.	210
B.2	ATMEL T89C51CC01: area di memoria destinata al bootloader.	212
B.3	ATMEL T89C51CC01: schematizzazione del processo di "boot" dopo il reset del sistema.	212
B.4	ATMEL T89C51CC01: schema di programmazione della flash memory.	214
C.1	ATMEL T89C51CC01: struttura generale del bit time ricavato dai parametri di configurazione.	218
D.1	OBD-II: schema del connettore di diagnostica on-board.	221
D.2	OBD-II: posizionamento abituale del connettore.	222

E.1	Tabella riepilogativa dei protocolli attualmente usati e prospettive di impiego del protocollo CAN in applicazioni automotive: PARTE I.	225
E.2	Tabella riepilogativa dei protocolli attualmente usati e prospettive di impiego del protocollo CAN in applicazioni automotive: PARTE II.	226
E.3	Tabella riepilogativa dei protocolli attualmente usati e prospettive di impiego del protocollo CAN in applicazioni automotive: PARTE III.	227
F.1	CAN <i>software</i> : schematizzazione di come il CAN <i>software</i> interpreta i campi “Start Bit” e “Lenght”.	230
G.1	CAN <i>software</i> : finestra di visualizzazione dei parametri di “WorkLoad”.	232
G.2	CAN <i>software</i> : parametri per il calcolo del “WorkLoad”.	233
G.3	CAN <i>software</i> : apertura della finestra temporale per il calcolo del parametro “WorkLoad” medio.	234

Elenco delle tabelle

1.1	Esemplificazione di alcuni protocolli.	7
1.2	Tecnologie attualmente in uso.	12
1.3	Elenco riepilogativo specifiche SAE relative al protocollo J1939.	18
1.4	CAL: identificatori associati ai servizi offerti dal CAL.	20
1.5	Assegnazione degli identificatori nel CANopen-Predefined Connection Set.	25
1.6	DeviceNet: struttura di un <i>Explicit Message</i>	26
3.1	Tabella riepilogativa dei ritardi introdotti sul segnale dai singoli componenti di un nodo CAN.	86
3.2	Tabella riepilogativa dei valori del baudrate espressi in funzione della lunghezza della linea.	95
4.1	Caratteristiche dei chip ATMEL della serie CANary a confronto.	104
4.2	Philips PCA82C250: elenco delle caratteristiche tecniche del transceiver.	115
4.3	Tabella comparativa caratteristiche RS-232, USB.	123
4.4	MODUSB1: elenco dei pin più significativi.	126
5.1	CANsoftware: comandi inviati.	132
5.2	CANanalyser: dettaglio della struttura del comando “iniarray[8]”.	134
5.3	MICROsoftware: dettaglio della struttura dei comandi inviati.	135
5.4	MICROsoftware: stima dei valori massimi di “bus load” che garantiscono l’acquisizione di dati in real-time senza perdita di informazioni.	142
5.5	MICROsoftware: comandi di bit timing inviati a CANsoftware.	150
5.6	CANsoftware: elenco ed interpretazione dei campi della vista “CCanDataView”.	154
5.7	Tabella riepilogativa valori campo parametri.	190
A.1	ATMEL T89C51CC01: configurazione dei pin.	208

C.1	ATMEL T89C51CC01: struttura del CAN Bit Timing register	
1.	216
C.2	ATMEL T89C51CC01: struttura del CAN Bit Timing register	
2.	217
C.3	ATMEL T89C51CC01: struttura del CAN Bit Timing register	
3.	218

Listati

3.1	Algoritmo di calcolo della sequenza CRC implementato nel protocollo CAN.	69
5.1	can_drv_it_overn()	146
5.2	MICRO <i>software</i> : parte dichiarativa della classe “CCanDATA”.	155
5.3	MICRO <i>software</i> : definizione del tipo “tCtrlCmd” utilizzato per definire la variabile membro “m_Control”.	159
5.4	MICRO <i>software</i> : implementazione della funzione di generazione del campo CRC del frame per mezzo della funzione “GenerateCRCsequence”.	161
5.5	CAN <i>software</i> : parte dichiarativa della classe “CCoda”.	164
5.6	CAN <i>software</i> : codice della procedura “Thread_ListenMode” tramite cui si implementa il Task2.	169